图文 112 当Broker获取到一条消息之后，他是如何存储这条消息的?

详情 评论

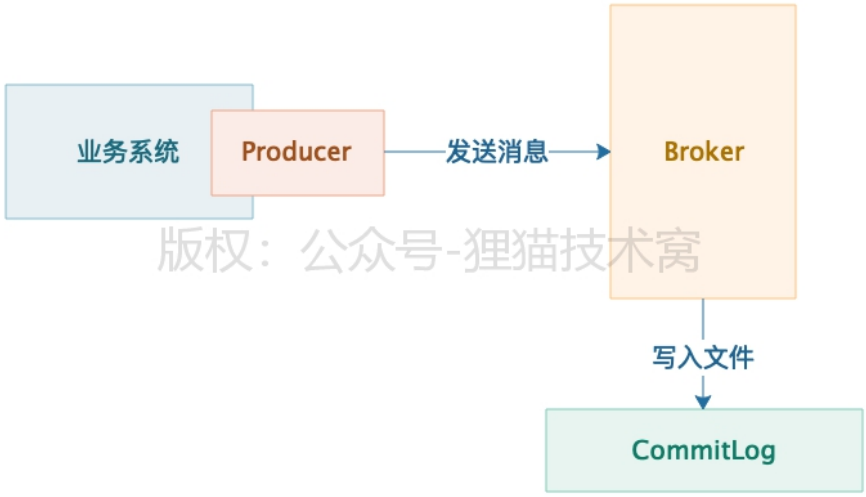
当Broker获取到一条消息之后，他是如何存储这条消息的?

继《从零开始带你成为JVM实战高手》后，阿里资深技术专家携新作再度出山，重磅推荐：

（点击下方蓝字试听）

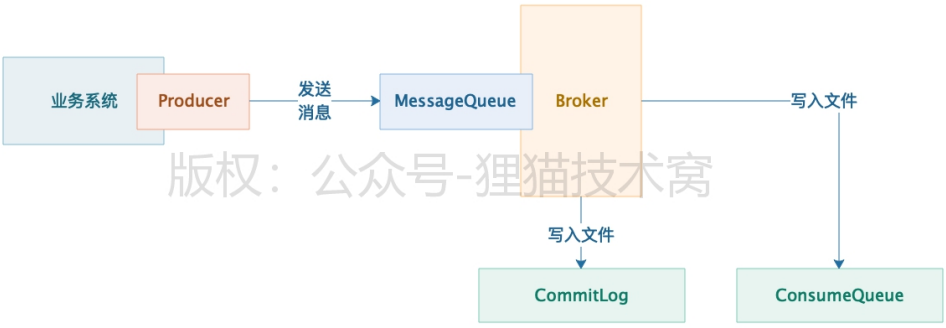
《从零开始带你成为MySQL实战优化高手》

其实之前我们就已经给大家在原理部分讲解过一些Broker收到消息之后的处理流程，简单来说，Broker通过Netty网络服务器获取到一条消息，接着就会把这条消息写入到一个CommitLog文件里去，一个Broker机器上就只有一个CommitLog文件，所有Topic的消息都会写入到一个文件里去，如下图所示。

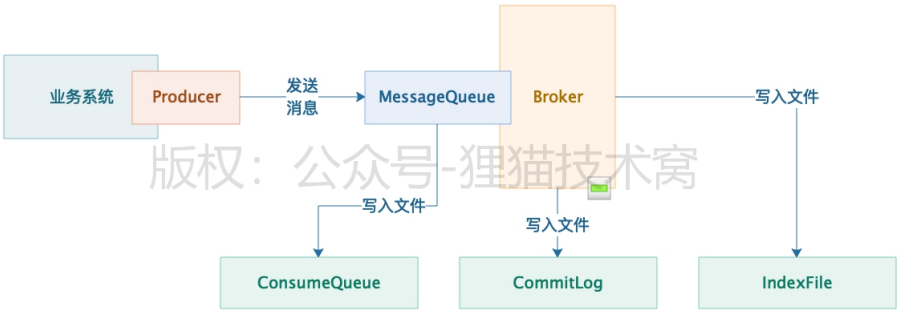


然后同时还会以异步的方式把消息写入到ConsumeQueue文件里去，因为一个Topic有多个MessageQueue，任何一条消息都是写入一个MessageQueue的，那个MessageQueue其实就是对应了一个ConsumeQueue文件

所以一条写入MessageQueue的消息，必然会异步进入对应的ConsumeQueue文件，如下图。



同时还会异步把消息写入一个IndexFile里，在里面主要就是把每条消息的key和消息在CommitLog中的offset偏移量做一个索引，这样后续如果要根据消息key从CommitLog文件里查询消息，就可以根据IndexFile的索引来了，如下图。



接着我们来一步一步分析一下他在这里写入这几个文件的一个流程

首先Broker收到一个消息之后，必然是先写入CommitLog文件的，那么这个CommitLog文件在磁盘上的目录结构大致如何呢？看下面

CommitLog文件的存储目录是在${ROCKETMQ_HOME}/store/commitlog下的，里面会有很多的CommitLog文件，每个文件默认是1GB大小，一个文件写满了就创建一个新的文件，文件名的话，就是文件中的第一个偏移量，如下面所示。文件名如果不足20位的话，就用0来补齐就可以了。

00000000000000000000
00000000003052631924

在把消息写入CommitLog文件的时候，会申请一个putMessageLock锁

也就是说，在Broker上写入消息到CommitLog文件的时候，都是串行的，不会让你并发的写入，并发写入文件必然会有数据错乱的问题，下面是源码片段。

```
protected final PutMessageLock putMessageLock;
putMessageLock.lock()
```

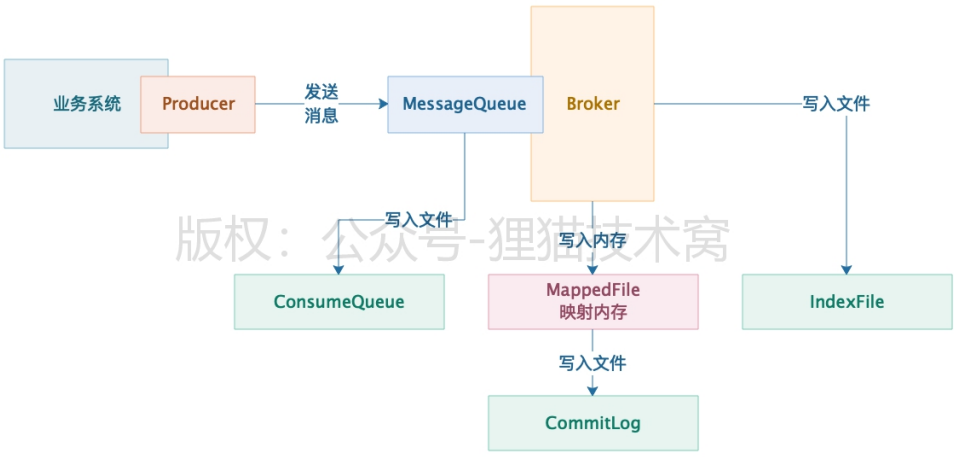
接着其实会对消息做出一通处理，包括设置消息的存储时间、创建全局唯一的消息ID、计算消息的总长度，然后会走一段很关键的源码，把消息写入到MappedFile里去，这个其实我们之前还讲解过里面的黑科技，看下面的源码。

```
ByteBuffer byteBuffer =
    writeBuffer != null ? writeBuffer.slice() : this.mappedByteBuffer.slice();

byteBuffer.position(currentPos);
AppendMessageResult result;

if (messageExt instanceof MessageExtBrokerInner) {
    result = cb.doAppend(
        this.getFileFromOffset(),
        byteBuffer, this.fileSize - currentPos,
        (MessageExtBrokerInner) messageExt);
} else if (messageExt instanceof MessageExtBatch) {
    result = cb.doAppend(
        this.getFileFromOffset(),
        byteBuffer,
        this.fileSize - currentPos,
        (MessageExtBatch) messageExt);
} else {
    return new AppendMessageResult(
        AppendMessageStatus.UNKNOWN_ERROR);
}
this.wrotePosition.addAndGet(result.getWroteBytes());
this.storeTimestamp = result.getStoreTimestamp();
return result;
```

上面源码片段中，其实最关键的是cb.doAppend()这行代码，这行代码其实是把消息追加到MappedFile映射的一块内存里去，并没有直接刷入磁盘中，如下图所示。



至于具体什么时候才会把内存里的数据刷入磁盘，其实要看我们配置的刷盘策略，这个我们后续会讲解，另外就是不管是同步刷盘还是异步刷盘，假设你配置了主从同步，一旦你写入完消息到CommitLog之后，接下来都会进行主从同步复制的。

那今天我们内容就讲到这里，其实到这里为止，就初步的讲完了Broker收到一条消息之后的处理流程了，先写入CommitLog中去。下一次我们讲解CommitLog的刷盘策略以及主从复制机制，然后接着再讲异步把消息写入ConsumeQueue和IndexFile里去。

End

重要说明：

- 如何提问：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑
- 如何加群：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

  具体加群方式，请参见目录菜单下的文档：《付费用户如何加群》（**购买后可见**）